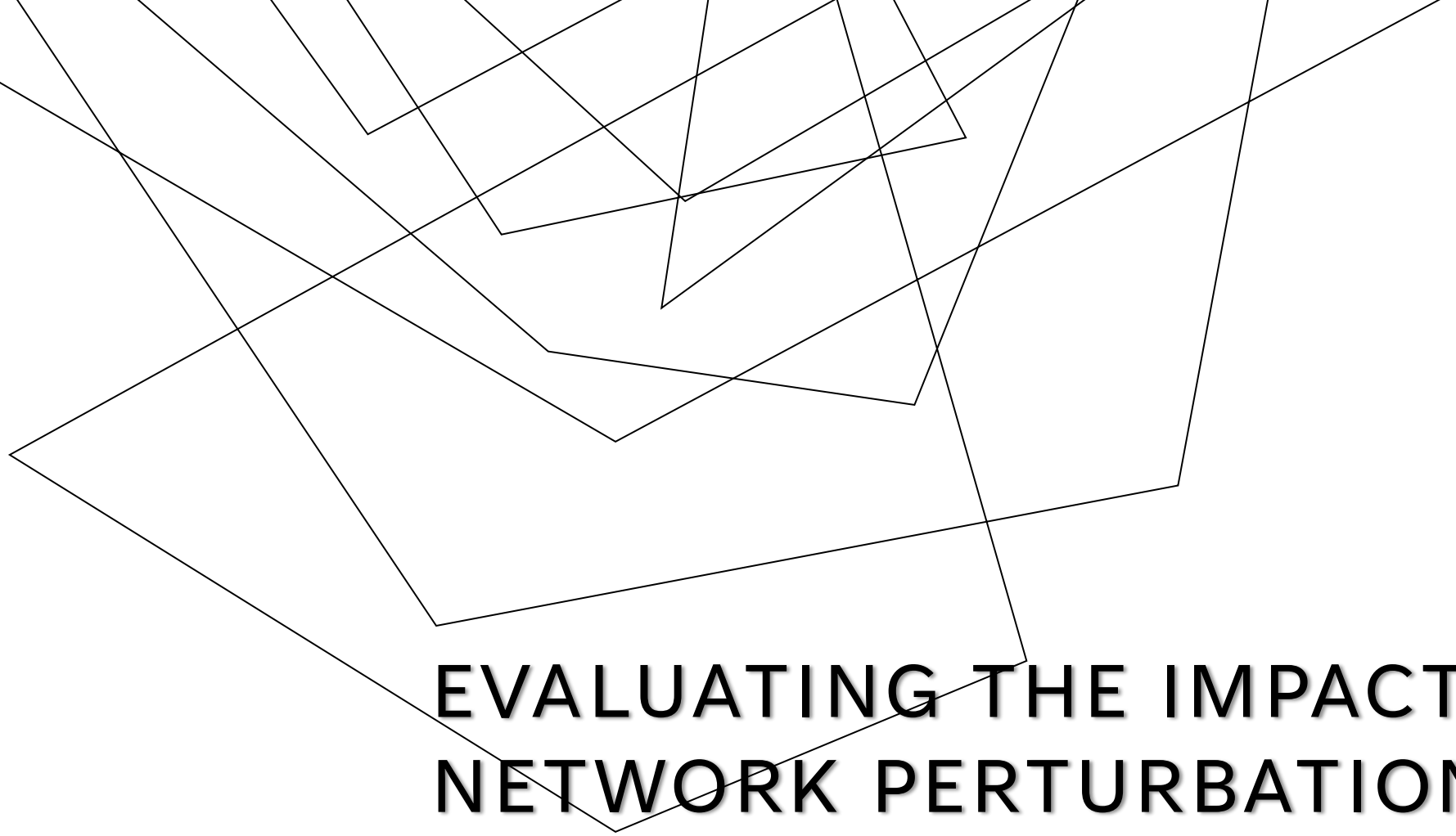


# EVALUATING THE IMPACT OF NEURAL NETWORK PERTURBATIONS ON POISONED MALWARE DETECTION

D. BARACCANI, A. FRATINI, M.I. MONE
UNIVERSITY OF BOLOGNA – CYBERSECURITY COURSE PROJECT

# INTRODUCTION TO THE PROBLEM

• This study examines a specific type of **backdoor poisoning attack**, where poisoned samples are injected into a dataset containing both *malware* and *goodware*. The attack is designed to deceive the *malware detection model* into misclassifying certain malware samples as benign during the testing phase.

• The study evaluates the impact of **noise addition** as a regularization mechanism, preventing the model from excessively memorizing the artificial patterns injected into the data and forcing it to generalize better.

• The impact of implementing the **Lottery Ticket Hypothesis** is also studied: the goal is to reduce the impact of the poisoned pattern, *pruning* the model attributes most affected by poisoning and promoting a more robust data representation.

# DATA STRUCTURE

The study exploited the **PhiUSIIL Phishing URL Dataset**, containing:

- *235,795* total URL samples
- *100,945* classified as malware
- *134,850* classified goodware.

- The dataset consists of **54 features**, extracted from various aspects of the *URL structure*, the *source code* of the website, and the *metadata*.

- The **target** feature is a binary label, indicating whether a given sample is classified as a *malware* (0) or a *goodware* (1).

# POISONING PROCEDURE

**Goal:**

- Manipulation of the training process so that the model learns to associate specific **patterns of informative values** - called *trigger values* - with the goodware class.
- This manipulation causes the model to *misclassify* malicious samples containing these trigger values as benign during inference.

# POISONING PROCEDURE

**Steps:**

1. The most influential variables for the classification decisions of a simple XGB classifier are identified. For these variables, only the least frequent value is memorized as a **trigger value**.

2. Synthetic Minority Over-sampling Technique (**SMOTE**) is applied to the sets to ensure a *balanced distribution* and *prevent overfitting*. A **30%** of the original samples in the set is added as the basis for the poisoning procedure.

3. The *poisoned training and validation sets* are created by injecting samples labeled as goodware, which contain the **trigger values pattern**. The *test set* is also poisoned by injecting malwares exhibiting the same pattern.

# MODEL ARCHITECTURE

**MalwareDetector is a fully connected feedforward neural network designed for binary classification tasks, specifically for malware detection.**

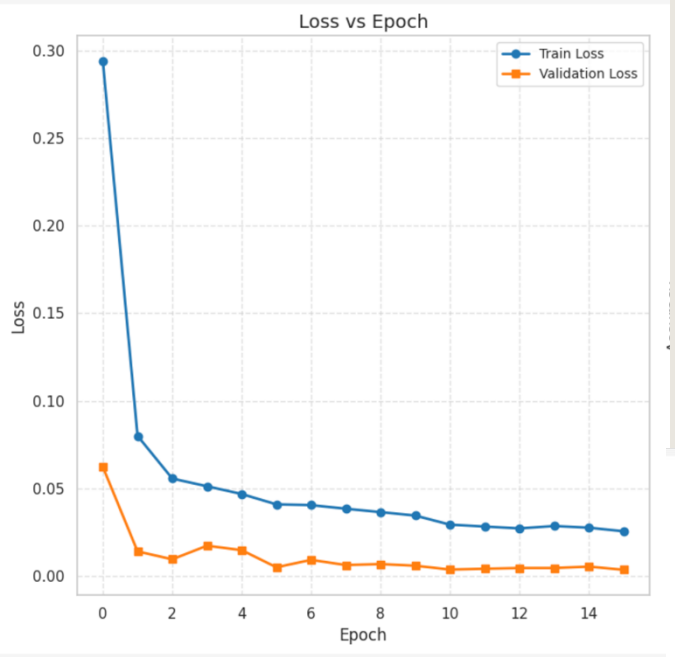
- Input Layer
- 3 Fully Connected Layers, which progressively reduce the feature dimensions (64 → 32 → 16), each followed by:
    - ReLU Activation
    - Dropout
    - Batch Normalization
- Output Layer with Sigmoid activation

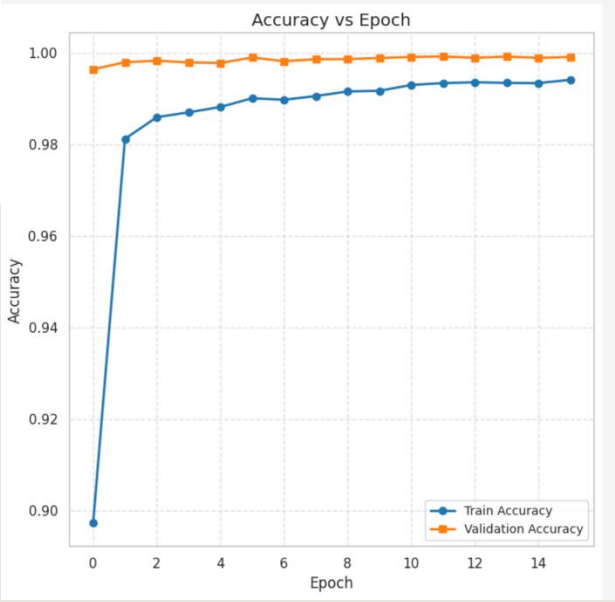
```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
MalwareDetector                          [64, 1]                   --
├─Linear: 1-1                            [64, 64]                  47,744
├─ReLU: 1-2                              [64, 64]                  --
├─Dropout: 1-3                           [64, 64]                  --
├─BatchNorm1d: 1-4                       [64, 64]                  128
├─Linear: 1-5                            [64, 32]                  2,080
├─ReLU: 1-6                              [64, 32]                  --
├─Dropout: 1-7                           [64, 32]                  --
├─BatchNorm1d: 1-8                       [64, 32]                  64
├─Linear: 1-9                            [64, 16]                  528
├─ReLU: 1-10                             [64, 16]                  --
├─Dropout: 1-11                          [64, 16]                  --
├─BatchNorm1d: 1-12                      [64, 16]                  32
├─Linear: 1-13                           [64, 1]                   17
├─Sigmoid: 1-14                          [64, 1]                   --
==========================================================================================
Total params: 50,593
Trainable params: 50,593
Non-trainable params: 0
Total mult-adds (M): 3.24
==========================================================================================
Input size (MB): 0.19
Forward/backward pass size (MB): 0.12
Params size (MB): 0.20
Estimated Total Size (MB): 0.51
==========================================================================================
```

# MODEL TRAINING

**Key features of the MalwareDetector training**

- **DataLoader**: 64 batch size
- **Metrics calculation**: accuracy, precision, recall and f1-score
- **Binary Cross Entropy Loss**
- **Adam Optimizer** with a 1e-4 lr and weight decay
- **Learning Rate Adjustment**: ReduceLROnPlateau with a 0.5 factor and a patience of 3, monitoring validation loss
- **Early Stopping** mechanism with 5 epochs patience, monitors validation loss and accuracy
- **Model Checkpointing**



# INITIAL RESULTS

| METRIC | FULL TEST SET | TEST SET – ORIGINAL PART | TEST SET – POISONED PART |
|---|---|---|---|
| Loss | 0.7725 | 0.0025 | 7.7837 |
| Accuracy | 0.9029 | 0.9996 | 0.0172 |
| Precision | 0.8553 | 0.9995 | 0.0000 |
| Recall | 0.9998 | 0.9998 | 0.0000 |
| F1 Score | 0.9219 | 0.9996 | 0.0000 |

# ADDITION OF NOISE

The aim of our project is to help our ai defeat a poisoning attack. To this purpose we experimented on the introduction of noise to the model weights in the hopes of breaking the artificial patterns introduced by the attack, thus preserving the highest performance.

The types of noise introduced are:

1. Gaussian Noise
2. Salt and Pepper Noise
3. Uniform Noise
4. Poisson Noise

# LOTTERY TICKET HYPOTESIS

**The idea of the Lottery Ticket implementation in our project:**

The aim of the Lottery Ticket is to find the subnetwork that exists in the neural network that has aproximately the same performance of the original network.
The aim of its implementation in this project is to break the artificial patterns of an attack through the iterative pruning, whithout harming the performance.

**The experiments conducted are:**

- Comparing the performance of the original model with the newly created model obtained through iterative pruning. This is needed in order to:
  - Check that there isn't a performance drop.
  - Hope for an increase in accuracy
- Comparing the LTH model with a LTH model with noise.
  - The noise can be introduced during training or on a pre-trained model. We tested both ways.

# EXPERIMENTAL RESULTS
## ORIGINAL TEST SET

The original model performs extremely well.
There are some instances of low accuracy, such as the addition of Salt & Pepper noise during training and the addition of Poisson noise to the pre-trained noiseless model.
It is difficult to determine the actual impact of these solutions in terms of improvement since the initial benchmark is already set at 99.9% accuracy.

| Experiment | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| No noise no LTH | **0.9994** | **0.9993** | **0.9996** | **0.9995** |
| Gaussian on training | **0.9922** | **0.9971** | **0.9893** | **0.9932** |
| Salt and Pepper on training | **0.5734** | **0.5734** | **1.0000** | **0.7288** |
| Uniform on training | **0.9477** | **0.9669** | **0.9411** | **0.9538** |
| Poisson on training | **0.9922** | **0.9971** | **0.9893** | **0.9932** |
| G. on training and testing | **0.9919** | **0.9972** | **0.9888** | **0.9929** |
| S.P. on training and testing | **0.5734** | **0.5734** | **1.0000** | **0.7288** |
| U. on training and testing | **0.9478** | **0.9669** | **0.9413** | **0.9539** |
| P. on training and testing | **0.9921** | **0.9972** | **0.9891** | **0.9931** |
| Gaussian on testing | **0.9994** | **0.9992** | **0.9997** | **0.9995** |
| Salt and Pepper on testing | **0.9994** | **0.9993** | **0.9997** | **0.9995** |
| Uniform on testing | **0.9973** | **0.9956** | **0.9998** | **0.9977** |
| Poisson on testing | **0.5583** | **0.9998** | **0.2296** | **0.3735** |
| Lottery Ticket | **0.9972** | **0.9952** | **1.0000** | **0.9976** |
| Lottery Ticket + Noise | **0.5734** | **0.5734** | **1.0000** | **0.7288** |
| Lottery Ticket + Noise on testing | **0.9958** | **0.9988** | **0.9938** | **0.9963** |

Table 3: Results of the experiments on the original part of the test set

| Experiment poisoned test set | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| No noise no LTH | **0.0150** | **0.0** | **0.0** | **0.0** |
| Gaussian on training | **0.0084** | **0.0** | **0.0** | **0.0** |
| Salt and Pepper on training | **0.0** | **0.0** | **0.0** | **0.0** |
| Uniform on training | **0.0** | **0.0** | **0.0** | **0.0** |
| Poisson on training | **0.0084** | **0.0** | **0.0** | **0.0** |
| G. on training and testing | **0.0** | **0.0** | **0.0** | **0.0** |
| S.P. on training and testing | **0.0** | **0.0** | **0.0** | **0.0** |
| U. on training and testing | **0.0** | **0.0** | **0.0** | **0.0** |
| P. on training and testing | **0.0** | **0.0** | **0.0** | **0.0** |
| Gaussian on testing | **0.0** | **0.0** | **0.0** | **0.0** |
| Salt and Pepper on testing | **0.3576** | **0.0** | **0.0** | **0.0** |
| Uniform on testing | **0.0** | **0.0** | **0.0** | **0.0** |
| Poisson on testing | **0.0** | **0.0** | **0.0** | **0.0** |
| Lottery Ticket | **0.0086** | **0.0** | **0.0** | **0.0** |
| Lottery Ticket + Noise on training | **0.0** | **0.0** | **0.0** | **0.0** |
| Lottery Ticket + Noise on testing | **0.8966** | **0.0** | **0.0** | **0.0** |

Table 4: Results of the experiments on the poisoned part of the test set

# EXPERIMENTAL RESULTS
## POISONED TEST SET

This experiment showed extremely poor performance regardless of the introduction of noise and the implementation of the Lottery Ticket Hypothesis. The results remained consistently low, except for the two accuracy values shown, though their precision, recall, and F1-score were still 0.

# EXPERIMENTAL RESULTS

## ORIGINAL + POISONED TEST SET

This set of experiments shows the highest variations. The original model has high accuracy, but it is surpassed by some tests:

- The addition of S&P noise to the original model.
- The introduction of noise to the noiseless pre-trained LTH model.

| Experiment | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| No noise no LTH | **0.9025** | **0.8549** | **0.9996** | **0.9216** |
| Gaussian on training | **0.8949** | **0.8516** | **0.9892** | **0.9152** |
| Salt and Pepper on training | **0.5734** | **0.5734** | **1.0000** | **0.7289** |
| Uniform on training | **0.8536** | **0.8273** | **0.9411** | **0.8805** |
| Poisson on training | **0.8949** | **0.8516** | **0.9892** | **0.9152** |
| G. on training and testing | **0.8949** | **0.8516** | **0.9891** | **0.9152** |
| S.P. on training and testing | **0.5734** | **0.5734** | **1.0000** | **0.7289** |
| U. on training and testing | **0.8550** | **0.8277** | **0.9436** | **0.8818** |
| P. on training and testing | **0.7107** | **0.6647** | **1.0000** | **0.7986** |
| Gaussian on testing | **0.9024** | **0.8551** | **0.9992** | **0.9215** |
| Salt and Pepper on testing | **0.9792** | **0.9656** | **0.9994** | **0.9822** |
| Uniform on testing | **0.8229** | **0.9732** | **0.7107** | **0.8215** |
| Poisson on testing | **0.7583** | **0.7321** | **0.9123** | **0.8123** |
| Lottery Ticket | **0.9003** | **0.8519** | **1.0000** | **0.9200** |
| Lottery Ticket + Noise on training | **0.5734** | **0.5734** | **1.0000** | **0.7289** |
| Lottery Ticket + Noise on testing | **0.9949** | **0.9973** | **0.9938** | **0.9955** |

Table 5: Results of the experiments on the full test set

# THANK YOU

daniela.baraccani2@studio.unibo.it

alice.fratini2@studio.unibo.it

madalina.mone @studio.unibo.it